ICE 整合Log4j

Ice 自带的日志很简陋，而且日志都是输出到一个文件中，这么多日志都输出到一个文件中，如果想查看某一个天的日志相当费劲。通常JAVA开发人员习惯了log4j这种强大的框架，如果现在使用ice自带的日志，感觉相当low。不过不要紧，Ice提供了与其它日志框架进行整合的接口，下面我们就来演示一下：

**1.pom中添加log4j依赖**

<dependency>

<groupId>log4j</groupId>

<artifactId>log4j</artifactId>

<version>1.2.17</version>

</dependency>

**2.resources 文件夹下添加log4j.properties配置文件，配置文件内容如下 :**

### 设置###

log4j.rootLogger = debug,stdout,D,E

### 输出信息到控制抬 ###

log4j.appender.stdout = org.apache.log4j.ConsoleAppender

log4j.appender.stdout.Target = System.out

log4j.appender.stdout.layout = org.apache.log4j.PatternLayout

log4j.appender.stdout.layout.ConversionPattern = [%-5p] %d{yyyy-MM-dd HH:mm:ss,SSS} method:%l%n%m%n

### 输出DEBUG 级别以上的日志到=D://logs/debug.log ###

log4j.appender.D = org.apache.log4j.DailyRollingFileAppender

log4j.appender.D.File = F://logs/debug.log

log4j.appender.D.Append = true

log4j.appender.D.Threshold = DEBUG

log4j.appender.D.layout = org.apache.log4j.PatternLayout

log4j.appender.D.layout.ConversionPattern = %-d{yyyy-MM-dd HH:mm:ss} [ %t:%r ] - [ %p ] %m%n

### 输出ERROR 级别以上的日志到=D://logs/error.log ###

log4j.appender.E = org.apache.log4j.DailyRollingFileAppender

log4j.appender.E.File =F://logs/error.log

log4j.appender.E.Append = true

log4j.appender.E.Threshold = ERROR

log4j.appender.E.layout = org.apache.log4j.PatternLayout

log4j.appender.E.layout.ConversionPattern = %-d{yyyy-MM-dd HH:mm:ss} [ %t:%r ] - [ %p ] %m%n

**3.自定义一个日志类LoggerI**

该类实现Ice.Logger接口，并在该类中获得log4j的日志logger,下面是代码

**public** **class** LoggerI **implements** Logger{

//在此获得log4j的日志logger

org.apache.log4j.Logger logger=org.apache.log4j.Logger.*getLogger*(LoggerI.**class**);

//相当于log4j的info

**public** **void** print(String message) {

logger.info(message);

}

//相当于log4j的trace

**public** **void** trace(String category, String message) {

logger.debug(category+":"+message);

}

//相当于log4j的warning

**public** **void** warning(String message) {

logger.warn(message);

}

//相当于log4j的error

**public** **void** error(String message) {

logger.error(message);

}

**public** String getPrefix() {

**return** "";

}

**public** Logger cloneWithPrefix(String prefix) {

**return** **new** LoggerI();

}

}

**4.自定义一个日志插件工厂类**

MyLoggerPluginFactoryI,该类实现PluginFactory接口.代码如下:

**public** **class** MyLoggerPluginFactoryI **implements** PluginFactory{

**public** Plugin create(Communicator communicator, String name, String[] args) {

**return** **new** LoggerPlugin(communicator, **new** LoggerI());

}

}

**5.通过Ice.Plugin.Logger 属性引入自定义的工厂类**

在ice的属性文件中添加:

*Ice.Plugin.Logger=com.projectice.utils.MyLoggerPluginFactoryI*

如果是icegrid,则在其xml文件中的属性结点下配置,如下图中的红色加粗部分

<icegrid>

<application name=*"app"*>

<properties id=*"MultiThreaded"*>

<property name=*"Ice.PrintStackTraces"* value=*"1"* />

<property name=*"Ice.Trace.Retry"* value=*"2"* />

<property name=*"Ice.Trace.Network"* value=*"1"* />

<property name=*"Ice.Trace.ThreadPool"* value=*"1"* />

<property name=*"Ice.Trace.Locator"* value=*"1"* />

**<property name=*"Ice.Plugin.Logger"* value=*"com.projectice.utils.MyLoggerPluginFactoryI"* />**

</properties>

..........其它略

这样，我们的日志整合就完成了，我们通过以下代码就可以按log4j的配置打印日志了

Ice.Logger logger=\_\_current.adapter.getCommunicator().getLogger();

logger.error("this is error log ");

logger.print("this is info log ");

logger.warning("this is warn log ");

logger.trace("","this is debug log ");